International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 3 Issue 11, November-2014

Design of Hardware Bus Monitoring Unit (HBMU) for Generic Power Management Unit (GPMU) in Soc

Anand Mohan
Assistant Professor ,
T John Institute Of Technology,
Department Of Ece,
Bangalore,

Saranya S Kumar,
Programmer Analyst,
CTS

***Abstract*-- The aim of the project is to design an efficient Hardware Bus Monitoring Unit (HBMU) for Generic Power Management Unit (GPMU) in SoC. Power management consists of a set of heterogenous tasks such as clock gating, power gating, reset and clock management and various other tasks.GPMU consists of various sub blocks in it. Each sub block is having separate function.System level work load estimator will estimate or monitor the workload and depending upon the values performance requests are done such as DVS, DFS, etc Bus tranasaction events and bus contention events are measured using HBMU.Latency and bandwidth are the typical parameters measured.Process monitor will monitor the processes inside the SoC and comparing the values stored in the power look up table power supply is varied. Dynamic Voltage Scaling (DVS),Adaptive Body Biasing (ABB),Dynamic Frequency Scaling(DFS),Adaptive Voltage Scaling(AVS) are the various adaptive power control techniques used.Using these techniques wastage of power is minimised and efficient use of power source is done.HBMU output will be given to PMU or CGU for efficient use of power.**

***Key words*- Generic Power Management Unit, Hardware Bus Monitoring Unit, System On Chip.**

I. INTRODUCTION

Power Management consists of a set of heterogeneous tasks. They are power gating, clock gating, reset and clock management, oscillator, PLL and clock generator control, monitoring of power events, management of advanced policies such as DVS,DFS,ABB and AVS. Monitoring of environmental changes are also done. The aim of this project is to design an efficient Hardware Bus Monitoring Unit (HBMU) for Generic Power Management Unit (GPMU) in SoC.
As the name indicates HBMU is a unit which monitors the different transaction and contention events occurring in a SoC. These values will be given to PMU or CGU to take necessary actions which is required for the low power consumption of SoC.

II. GENERIC POWER MANAGEMENT UNIT(GPMU)

The following block diagram Fig 1 shows the functional level description of GPMU.

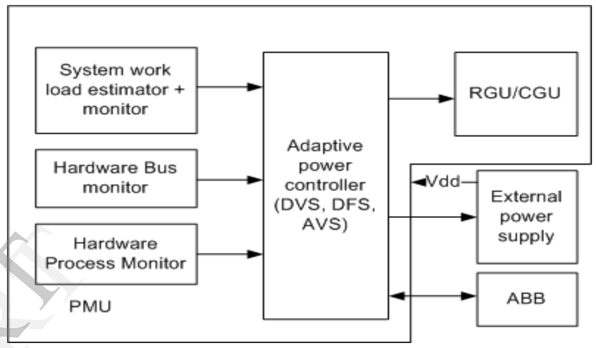


Fig 1 GPMU

The various sub blocks present in the GPMU are System work load estimator and monitor, Hardware Bus monitor, Hardware Process monitor, Adaptive Power Controller and RGU/CGU.

A. System Work Load Estimator And Monitor

It will estimate or monitor the system workload to generate a performance request. For monitoring open loop or closed loop implementation of power management scheme are used. In open loop, static approach is used. Lookup table approach is used typically. Closed loop is suitable for dynamic power mangement.

B. Hardware Bus Monitor

Bus transaction events and bus contention events are measured using Bus Monitors (BM).It consists of Master ports and Slave ports connected to system. Each master port and slave port are connected to the BM using distributors. This is done in order to reduce the number of BM. Latency and bandwidth are the typical parameters measured. These parameters will be send to PMU and CGU(under software control).

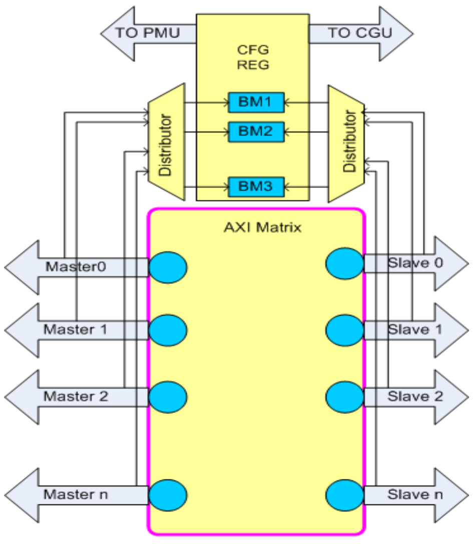


Fig 2 Hardware Bus Monitor

C. Hardware Process Monitor

It will monitor the processes in the SoC and these values will be send to the PMU. Using the values present in the power look up table present inside the PMU, the power supply is varied and given to the SoC so that power is efficiently used and wastage is reduced.

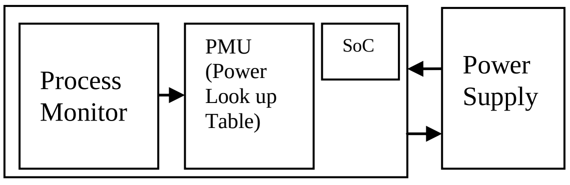


Fig 3 Hardware Process Monitor

D. Adaptive Power Controller

It consists of DVS, ABB, DFS, AVS. Different applications that run on a processor may have varying performance requirements. These applications can be effectively run at different voltages in various modes of the chip operation. Appropriate voltage can determined through a frequency-based voltage lookup table. PMIC can be directed by the power controller to provide the appropriate supply. Level-shifters are needed on signals that cross voltage boundaries. Level-shifters adapt to shifting from a scaled island scenario to a fixed voltage level or from a fixed voltage level to a scaled scenario. DVS is also known as dynamic voltage and frequency scaling (DVFS).

Active back bias (ABB) voltage, applied to wells of N-MOS and P-MOS transistors, is used to set the threshold voltages and leakage currents precisely in order to improve speed and at the same time control device sub-threshold leakage. The active back bias applies a voltage to the well of devices and this voltage can be generated by a PMIC. If the leakage increases with age, temperature, or other conditions, changes in bias supply can be used to compensate. Compensation is required for process variations also; ABB is likely to become a necessity at 45nm and below.

Dynamic Frequency Scaling as the name indicates the frequency is varied in order to speed up or speed down the processor depending upon the situation.

From a power management concept standpoint, AVS is similar to DVS. However, unlike DVS which uses table lookup, AVS is closed loop system and power controller interfaces with a monitor in the scaled block to determine frequency needs and then directs the PMIC to provide appropriate voltage. Level shifting needs are similar to that of a DVS scheme. Design and verification implications are similar to that of DVS scheme.

E. RGU/CGU

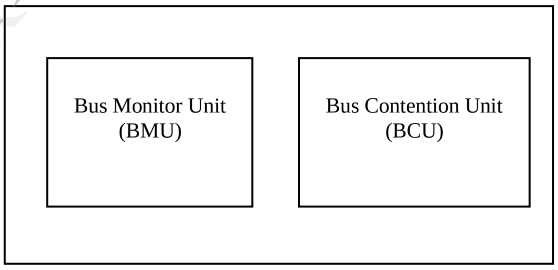
RGU stands for Flexible Reset Generation Unit. It is used to control the reset the individual modules.CGU stands for Flexible Clock Generation Unit. It is used to control the clock of individual modules.

III. DESIGN OF HBMU

Hardware Bus Monitoring Unit (HBMU) consists of three main units. They are

1. Bus Monitor Unit (BMU)
2. Bus Contention Unit (BCU)

Bus transaction events and bus contention events are measured using BMU and BCU respectively. Latency and bandwidth are the typical parameters measured. These parameters will be given to PMU and CGU.



Block Diagram of Hardware Bus Monitoring Unit (HBMU)

A. Bus Monitor Unit (Bmu)

The BM collects information in regard to bus-transactions. The measurements of the BM are shown in Table 1, which are the total transaction count, the total transfer size (in bytes), the read latency distribution, the write latency distribution, the valid cycle count, and the busy cycle count. In addition, the BM has four global clock counters. Each of them is a 64-bit counter. The global clock counter is used to measure total clock counts for a specified duration. The read/write latency has wide distribution, such as from a couple of cycles to hundreds of cycles. So, it is not effective to have dedicated counters for each latency cycle. Therefore, we use an interval based counting method to count the read and write latency effectively.

We can select eight intervals for read latency and four intervals for write latency by software setting. A set of 32-bit counters (eight counters for read and four counters for write) increments its value once for each occurrence of the transaction within the specified latency interval. The BM is generally starts or stops monitoring by its register setting. In addition to this basic mechanism, the monitor could be activated by the control signal generated by another monitor unit or specified bus-transaction events in the bus to support various measuring requirements. The BM also has a feature to generate an interruption when specified events occur.

Parameter	Description
Total transaction count (read/write)	Number of transaction generated on the bus
Totaltransfer size (read/write)	Transferred data size on the bus
Read latency distribution	Each read latency counter is incremented by one for each occurrence of the event within the specified latency interval
Write latency distribution	Each write latency counter is incremented by one for each occurrence of the event within the specified latency interval
Global clock count	General purpose clock counter
Valid count (read/write)	Number of clock cycles to be required for transferring actual data
Busy count (read/write)	Number of clock cycles to be required for completing transaction (i.e., transaction length)

Table 1. Features of Bus Monitor

A master can generate transactions to several destinations (slaves) through the AXI interconnect. When we try to observe bus-transactions on a master bus, it would be useful if the BM could classify the destination of each bus-transaction. Therefore, the BM was designed to contain three performance counter (PC) sets for distinguishing bus-transaction events according to the address ranges. The BM has three address range divisions. Each PC unit, denoted as S0_PC, S1_PC, and S2_PC in Fig. 3.2.1, measures bus transaction events within the specified address range. An architect can modify an address range for observing the specified bus-transaction events by setting the control register of the bus monitor. Similarly, a slave can receive transactions from multiple masters through the AXI interconnect. When we intend to monitor bus-transactions on a slave bus, it would be useful if the BM could identify the source of the bus-transactions. The BM has a set of ID registers for this purpose. When the transaction ID registers are set and enabled, the BM only counts the transactions that have the same ID as we set in the ID register. Each PC can store the total transaction count, the total transfer size, the read latency distribution, the write latency distribution, the valid cycle count and the busy cycle count. Each transaction is caught by the transaction detection module, and the information is sent to the appropriate PC unit by the address decoder. Four global clock counters are located as an independent module, not included in the PC. The BM has a control register with an APB interface and a set of registers to setup monitoring conditions and to fetch measured value.

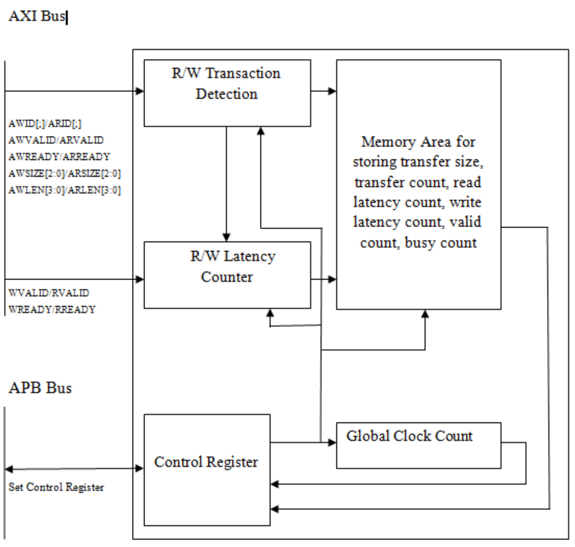


Fig 4 Bus Monitor Unit

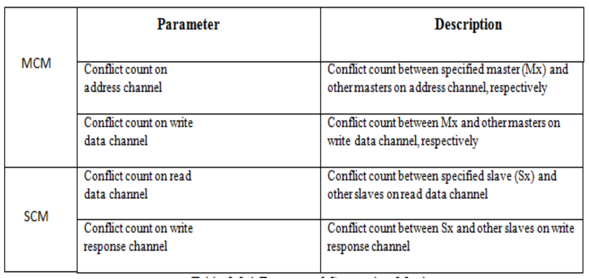
B. Bus Contention Unit

The bus contention occurs when multiple master (slave) devices are trying to transfer data to the same destination at the same time. It consists of two units.

1.Master Contention Monitor (MCM)

2. Slave Contention Monitor (SCM)

In the process of transferring data through the AXI inter connect; one device can have the right to send data through the bus based on the arbitration scheme. Other devices should wait until the device completes the transaction. We defined this status as a contention, and designed the BCU to measure the number of contentions quantitatively. Each master can transmit its signals to the slave after granting ownership in the master port of the AXI inter connect. Similarly, each slave can send signals to the master after accepting ownership in the slave port of the AXI inter connect. The specific features of the MCM and the SCM in 6 x 2 AXI interconnect are described in Table 2 The MCM measures conflict counts for an address channel and a write data channel, and the SCM counts the contention events for the read data channel and the write response channel. As the number of masters and slaves increases, the conflict case between masters and between slaves increases rapidly. In order to measure conflicts on master ports and slave ports in the AXI interconnect, the CM should know the transaction information of all accessible devices to the master (slave) port, such as which device occupies the bus, and which devices are waiting for completing current transaction.



	Parameter	Description
MCM	Conflict count on address channel	Conflict count between specified master (Mx) and othermasters on address channel, respectively
	Conflict count on write data channel	Conflict count between Mx and othermasters on write data channel, respectively
SCM	Conflict count on read data channel	Conflict count between specified slave (Sx) and otherslaves onread data channel
	Conflict count on write response channel	Conflict count between Sx and other slaves on write response channel

Table 2 Features of Contention Monitor

To measure this information, the MCM should access the internal signal of the router and arbitration signals of the master port in the internal AXI interconnect in addition to the AXI signals of the slave bus. The SCM should also access the internal signals of the router and arbitration signals of the slave port in the internal AXI interconnect in addition to the AXI signals of the master bus. The monitor unit detects a conflict when a transaction initiates by checking whether the transaction is generated by other devices except current bus-occupied device or not. Fig. 5 shows the structure of the MCM connected to an AXI interconnect with six slave ports (connected to six masters). The structure of the SCM is similar to that of the MCM. The SCM measures the conflict counts between a user-defined slave (i.e., Slave X) and other slaves on read data channel and write response channel. Fig. 6 shows the structure of the SCM connected to an AXI interconnect with two master ports (connected to two slaves). The structure of MCM and SCM should be modified slightly according to the number of master devices and slave devices connected to the AXI interconnect because the connected routing signals and the number of counter depends on the number of masterports and slave ports in the AXI interconnect.

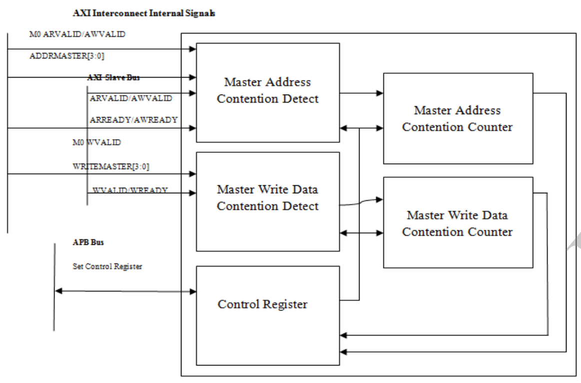

AXI Interconnect Internal Signals
M0 ARVALID/AWVALID
ADDRMASTER[3:0]
AXI Slave Bus
ARVALID/AWVALID
ARREADY/AWREADY
M0 WVALID
WRITEMASTER[3:0]
WVALID/WREADY
APB Bus
Set Control Register
Master Address Contention Detect
Master Address Contention Counter
Master Write Data Contention Detect
Master Write Data Contention Counter
Control Register

Fig 5 Master Contention Monitor

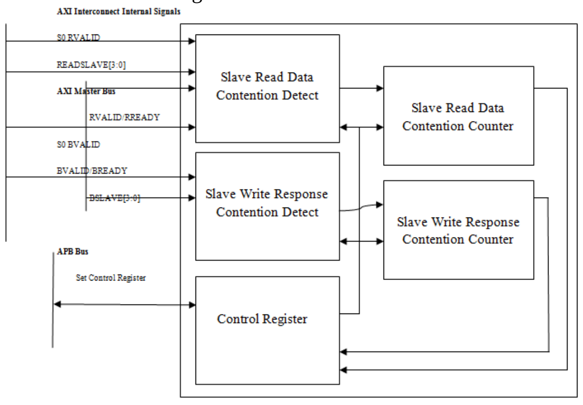

AXI Interconnect Internal Signals
S0 RVALID
READSLAVE[3:0]
AXI Master Bus
RVALID/RREADY
S0 BVALID
BVALID/BREADY
BSLAVE[3:0]
APB Bus
Set Control Register
Slave Read Data Contention Detect
Slave Read Data Contention Counter
Slave Write Response Contention Detect
Slave Write Response Contention Counter
Control Register

Fig 6 Slave Contention Monitor

IV. SIMULATION RESULTS

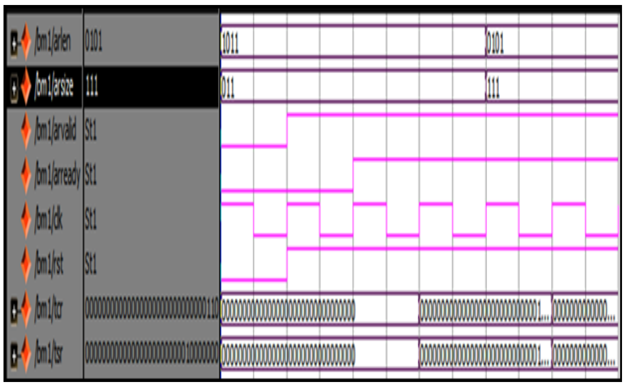


Fig 7 Simulation Result for Read Transaction Count and Size

Only when arvalid and arready signals are high, the read transaction count and size will be found out. Otherwise the process will not take place. Depending upon the values given for arlen and arsize the corresponding values of tcr and tsr will be displayed.

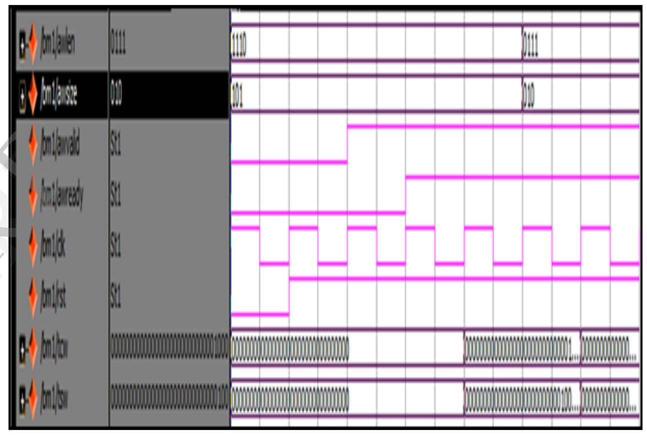


Fig 8 Simulation Result for Write Transaction Count and Size

Similarly for getting write transaction count and size awvalid and awready signals must be high. Depending upon the values given for awlen and awsize the values for tcr and tsr will be displayed.

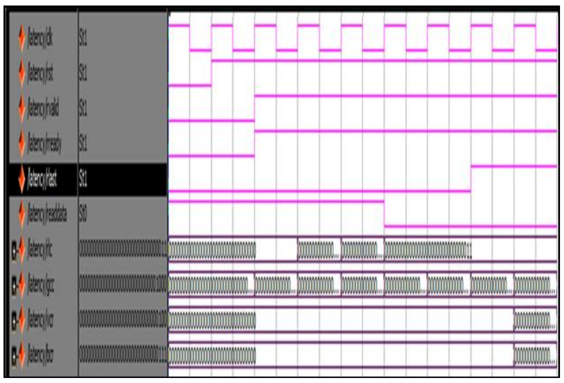


Fig 9 Simulation Result for Read Latency Count, Valid Count and Busy Count

The signals rvalid and rready must be high for the process to start. Variable rlc represents read latency count, it gets incremented only when the readdata signal is high. The total clock cycles required for the entire process is stored in gcc variable. It gets incremented till the signal rlast becomes high. The variable bcr represents the busy count for read transaction ie the total clock cycles required for the entire process. Valid count vcr is calculated using gcc and bcr.

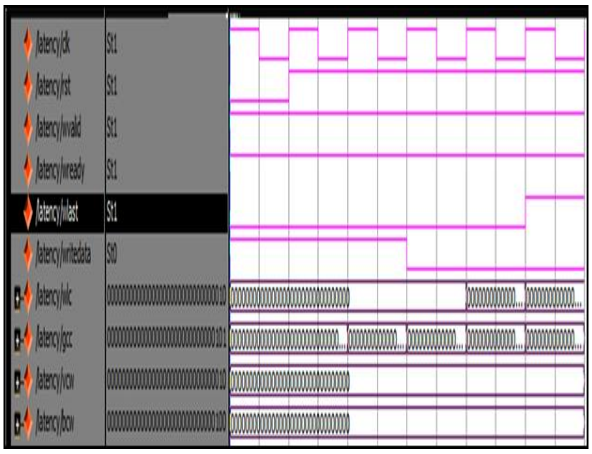


Fig 10 Simulation Result for Write Latency Count, Valid Count and Busy Count

Similarly for write transactions, the signals wvalid and wready must be high for the process to start. Variable wlc represents write latency count, it gets incremented only when the writedata signal is low. The total clock cycles required for the entire process is stored in gcc variable. It gets incremented till the signal wlast becomes high. The variable bcw represents the busy count for write transaction ie the total clock cycles required for the entire process. Valid count vcw is calculated using gcc and bcr.

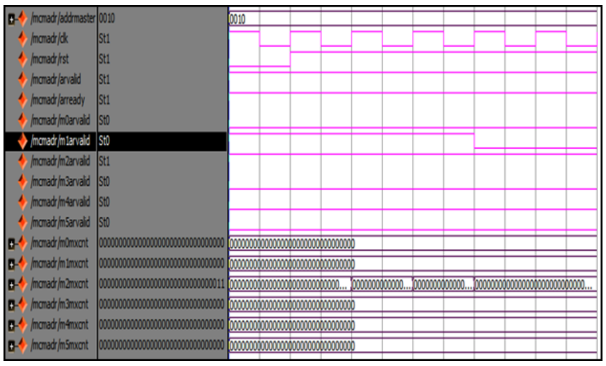


Fig 11 Simulation Result for MCM conflict count on Address channel (read)

The signals arvalid and arready must be high for the process to start. Depending upon the addrmaster signal the master read signal can be known. So if the signals m0arvalid,m1arvalid,etc except for the master read signal gets high the contention value increments. Here the master read signal is m2arvalid since the addrmaster is ‘0010’. So if the values of signals other than m2arvalid becomes high m2mxcnt ie contention value gets incremented.

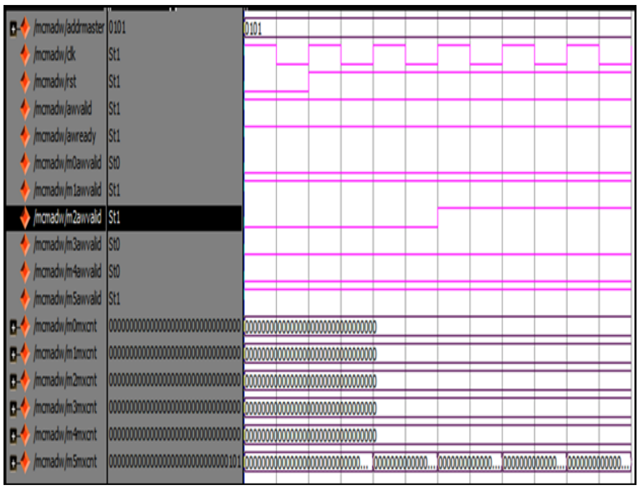


Fig 12 Simulation Result for MCM conflict count on Address channel (write)

The signals awvalid and awready must be high for the process to start. Depending upon the addrmaster signal the master write signal can be known. So if the signals m0awvalid,m1awvalid,etc except for the master write signal gets high the contention value increments. Here the master write signal is m5awvalid since the addrmaster is ‘0101’. So if the values of signals other than m5arvalid becomes high m5mxcnt ie contention value gets incremented.

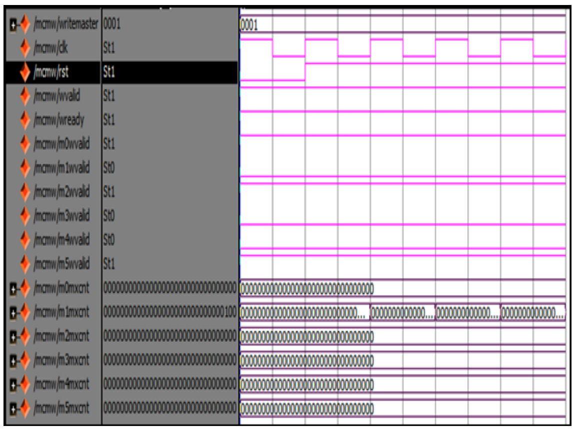


Fig 13 Simulation Result for MCM conflict count on Data channel (write)

The signals wvalid and wready must be high for the process to start. Depending upon the writemaster signal the master write signal can be known. So if the signals m0wvalid,m1wvalid,etc except for the master write signal gets high the contention value increments. Here the master write signal is m1wvalid since the addrmaster is ‘0001’. So if the values of signals other than m1wvalid becomes high m1mxcnt ie contention value gets incremented.

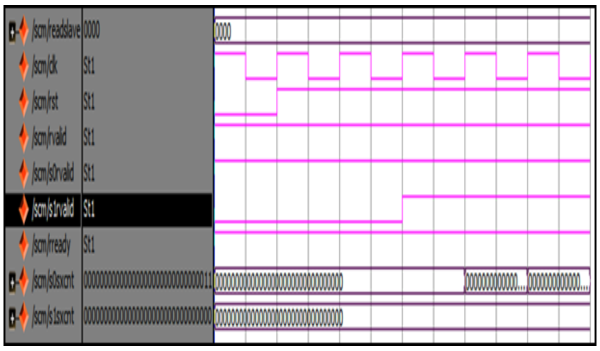


Fig 14 Simulation Result for SCM conflict count on Data channel (read)

The signals rvalid and rready must be high for the process to start. Depending upon the readslave signal the slave master read signal can be known. So if the signals s0rvalid or s1rvalid except for the master write signal gets high the contention value increments. Here the slave master read signal is s0rvalid since the readslave is ‘0000’. So if the values of signals other than s0rvalid becomes high, s0sxcnt ie contention value gets incremented.

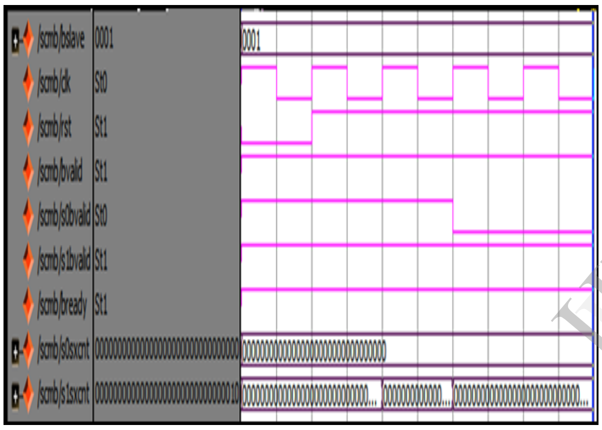


Fig 15 Simulation Result for SCM conflict count on Data channel (write response)

The signals bvalid and bready must be high for the process to start. Depending upon the bslave signal the slave master write response signal can be known. So if the signals s0bvalid or s1bvalid except for the slave master write response signal gets high the contention value increments. Here the slave master write response signal is s1bvalid since the bslave is ‘0001’. So if the values of signals other than s1bvalid becomes high, s1sxcnt ie contention value gets incremented.

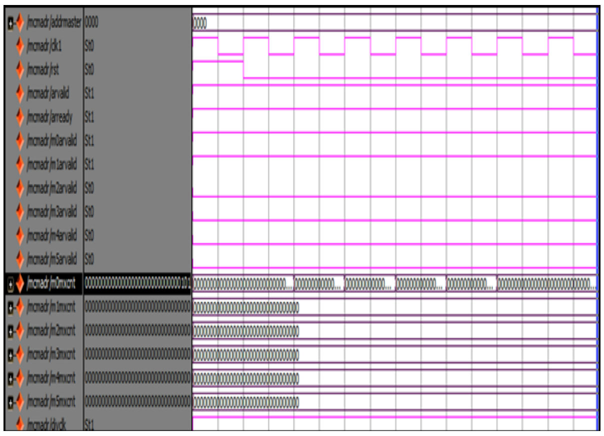


Fig 16 Simulation Result for MCM conflict count on Address channel (read) [Clock Gating]

The signals arvalid and arready must be high for the process to start. Depending upon the addrmaster signal the master read signal can be known. So if the signals m0arvalid,m1arvalid,etc except for the master read signal gets high the contention value increments. Here the master read signal is m0arvalid since the addrmaster is ‘0000’. So if the values of signals other than m0arvalid becomes high m0mxcnt ie contention value gets incremented. The program is designed in such a way that if any of the contention value is greater than 32’d4, clock gating is enabled.

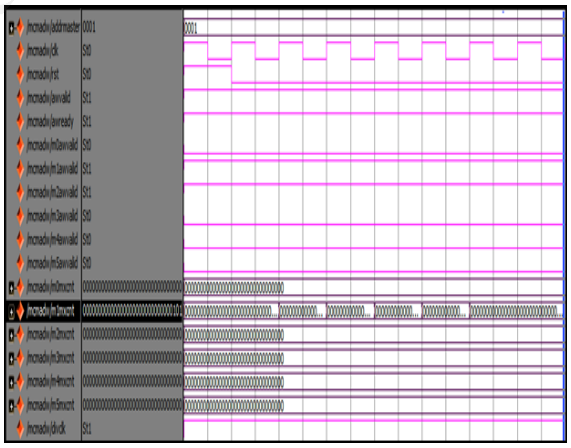


Fig 17 Simulation Result for MCM conflict count on Address channel (write) [Clock Gating]

The signals awvalid and awready must be high for the process to start. Depending upon the addrmaster signal the master write signal can be known. So if the signals m0awvalid,m1awvalid,etc except for the master write signal gets high the contention value increments. Here the master write signal is m1awvalid since the addrmaster is ‘0001’. So if the values of signals other than m1awvalid becomes high m1mxcnt ie contention value gets incremented. The program is designed in such a way that if any of the contention value is greater than 32’d4, clock gating is enabled.

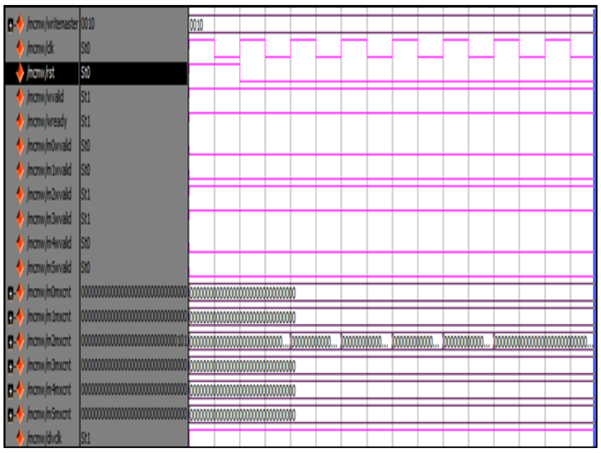


Fig 18 Simulation Result for MCM conflict count on Data channel (write) [Clock Gating]

The signals wvalid and wready must be high for the process to start. Depending upon the writemaster signal the master write signal can be known. So if the signals m0wvalid,m1wvalid,etc except for the master write signal gets high the contention value increments. Here the master write signal is m2wvalid since the addrmaster is '0010'. So if the values of signals other than m2wvalid becomes high m2mxcnt ie contention value gets incremented. The program is designed in such a way that if any of the contention value is greater than 32'd4, clock gating is enabled.

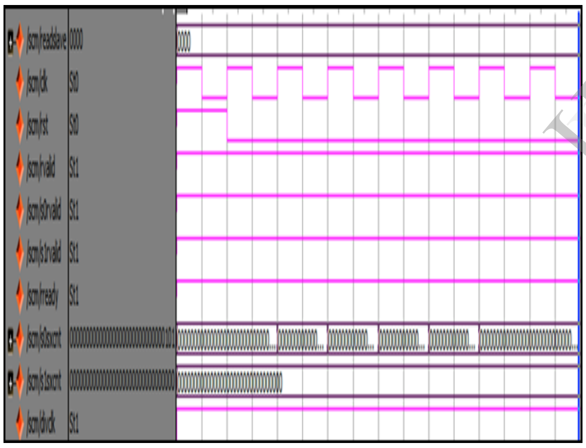


Fig 19 Simulation Result for SCM conflict count on Data channel (read) [Clock Gating]

The signals rvalid and rready must be high for the process to start. Depending upon the readslave signal the slave master read signal can be known. So if the signals s0rvalid or s1rvalid except for the master write signal gets high the contention value increments. Here the slave master read signal is s0rvalid since the readslave is '0000'. So if the values of signals other than s0rvalid becomes high, s0sxcnt ie contention value gets incremented. The program is designed in such a way that if any of the contention value is greater than 32'd4, clock gating is enabled.

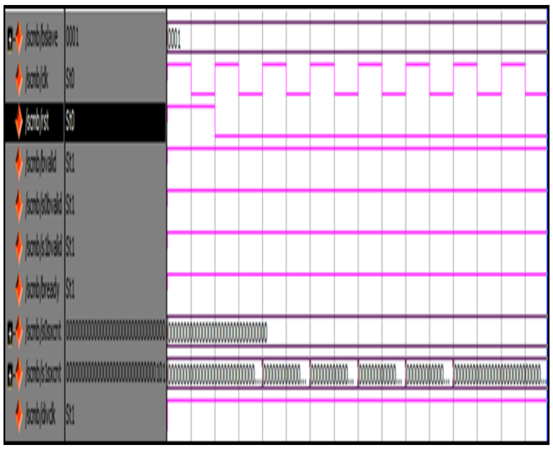


Fig 20 Simulation Result for SCM conflict count on Data channel (write response) [Clock Gating]

The signals bvalid and bready must be high for the process to start. Depending upon the bslave signal the slave master write response signal can be known. So if the signals s0bvalid or s1bvalid except for the slave master write response signal gets high the contention value increments. Here the slave master write response signal is s1bvalid since the bslave is '0001'. So if the values of signals other than s1bvalid becomes high, s1sxcnt ie contention value gets incremented. The program is designed in such a way that if any of the contention value is greater than 32'd4, clock gating is enabled.

DC COMPILER REPORTS

Table 3 Area Calculation

	Bus Monitor	Latency	MCM	SCM
No. of Ports	148	234	206	74
No. of Nets	202	607	617	207
No. of Cells	80	440	417	135
No. of References	11	14	16	8
Combinational Area	411.033607		5268.787273	1728.921625
Non Combinational Area	838.656013	7225.344116	6193.152100	2064.384033
Net Interconnect Area	58.733937	846.444436	742.256010	200.099615
Total Cell Area	1249.689621	13222.195323	11461.939373	3793.305658
Total Area	1308.423558	14068.639759	12204.195383	3993.405274

Table 4 Power Calculation

	Bus Monitor	Latency	MCM	SCM
Global Operating Voltage	0.7	0.7	0.7	0.7
Voltage Units	1V	1V	1V	1V
Capacitance Units	1.000000ff	1.000000ff	1.000000 ff	1.000000ff
Time Units	1ns	1ns	1ns	1ns
Dynamic Power Units	1uW	1uW	1uW	1uW
Leakage Power Units	1pW	1pW	1pW	1pW
Cell Internal Power	6.3077 uW (62%)	13.9882 uW (53%)	6.1571 uW (61%)	1.1773 uW (68%)
Net Switching Power	3.8079 uW (38%)	12.5578 uW (47%)	3.9965 uW (39%)	553.7650 nW (32%)
Total Dynamic Power	10.1156 uW (100%)	26.5460 uW (100%)	10.1536 uW (100%)	1.7311 uW (100%)
Cell Leakage Power	3.2370 uW	41.4315 uW	39.3466 uW	13.0791 uW

IMPLEMENTATION

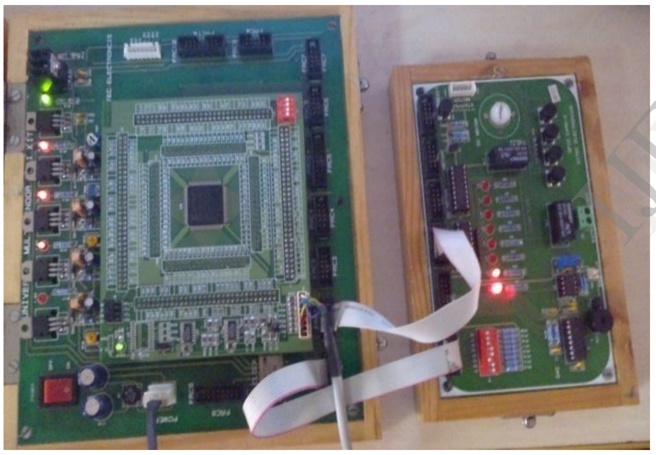


Fig 21 Implementation in Spartan 2 FPGA kit

The two units 1. Bus Monitoring Unit (BMU) and 2. Bus Contention Unit (BCU) have been implemented in Spartan 2 FPGA kits using Xilinx ISE software.

V. CONCLUSION

The overall architecture of Hardware Bus Monitoring Unit (HBMU) in Generic Power Management Unit (GPMU) have been studied. Different blocks of HBMU have been designed. The design of Bus Monitoring Unit (BMU) and Bus Contention Unit (BCU) have been done and simulated using ModelSim Simulator using behavioral level programming in Verilog.The implementation of these units have been done using Xilinx ISE in Spartan 2 and Spartan 3E.Depending upon the values of contention unit clock gating is done.
Various other blocks of GPMU can be done such as System Work Load Estimator, Hardware Process Monitor etc. and can be implemented into a single unit. Area and power can be efficiently managed in an SoC.

REFERENCES

1. Younghyun Kim, Sangyoung Park, Youngjin Cho, Naehyuck Chang, “System-Level Online Power Estimation Using an On-Chip Bus Performance Monitoring Unit”, *IEEE Transactions On Computer-Aided Design Of Integrated Circuits And Systems*, vol. 30, no. 11,pp. 1585-1598, Nov 2011.
2. Yi-Hao, Chang, Ing-Jer Huang, “A Performance Monitoring Tool Suite for Software and SoC On-Chip Bus”, 2011.
3. Hyun-min Kyung, Gi-ho Park, Jong Wook Kwak, Tae-jin Kim, Sung-Bae Park, “Design and implementation of Performance Analysis Unit (PAU) for AXI-based multi-core System on Chip (SOC)”, *Science Direct Microprocessors and Microsystems*, vol. 34, pp. 102–116, 2010.
4. Chi-Hung Lin, Chung-Fu Kao, and Ing-Jer Huang, “Configurable On-Chip Real Time Bus Tracer”, 2010.
5. Luca Benini, Giovanni De Micheli, “Network On Chips: Technology and Tools”, *Morgan Kauffman Publishers*.
6. AMBA AXI Protocol Specification v1.0, ARM, 2003.